CS/EECE 433 Fall 2006
E. Angel

Project Three: Three Dimensional Maze Game

The goal of this project is to create a three-dimensional maze that a user can explore interactively. The project can separated into two parts: (1) creation of a maze and (2) creating the interactive interface to navigate through the maze.

You can get started by first creating a two-dimensional maze in the plane y=0. Such a maze can be looked at as an array of N*M square cells, where N is the number of rows and M is the number of columns in the maze. Each cell has the property that one to three of its sides (or walls) are present. If two walls are present there is a path into the cell and a path leaving the cell. If three walls are present, the cell is a dead end. It is also possible to have a cell with a single wall which would indicate that there are multiple ways to leave the cell. Note: a maze can be represented by a graph in which each node represents a cell and each branch is a connection between two adjacent cells, i.e. the absence of a wall between the cells.

The particular type of maze you are to construct is one in which every cell is connected to every other cell and there is exactly one simple path between any pair of cells. One method to construct such a maze is to start with an array of cells, each of which has all four walls present. We then remove various walls to construct the desired maze.

Consider the following algorithm. Initially all cells are said to be unvisited. We start we start by selecting (visiting) a random cell. We connect this cell to one of its neighbors by randomly selecting one of its unvisited neighbors. We remove the wall between the two cells to connect them. If any of the other three neighbors has not been visited, we put their identifiers on a stack. If we reach a dead end (a cell all of whose neighbors has been visited), we pop an unvisited cell off the stack. When all cells have been visited, the stack is empty and we have a maze. We can then remove two pieces of the outside wall to create an entrance and an exit to the maze. Although there are many other ways to construct a maze, this type of maze will have only one path between the entrance and exit that does not contain loops.

You can convert your two dimensional maze to a three-dimensional on by making each line that separates two cells into a wall, i.e. a rectangular polygon. You can then use two large rectangular polygons to make a floor and a ceiling for the whole maze. As in the first assignment, you should add an entrance and exit.

The initial display should position the user in front of the entrance. The right mouse button should turn the user to the right, the left button to the left. You can program the mouse either to turn a fixed amount with each click (90 degrees) or to turn continuously as the mouse is held down. The middle button should move the user forward, either a single click moving the user forward one cell or through continuous motion as long as the mouse button is held down.

You should not allow the user to move through a wall.

Viewing should be in perspective.

Termination of the program or restarting can be done via the keyboard.

Backward motion may be added, possibly via combining the mouse and a button, e.g. shift + middle mouse.

You should use double buffering as it is necessary to have a smooth display.

Note there are many possibilities for adding extras such as building up a map for the user of the parts of the maze already explored and displaying this map on the screen. You might also add shading and/or textures to the walls.

Note it is possible to use a simple extension of the original maze algorithm to create a true three-dimensional multi-story maze. You can do this for extra credit but note that since there will now be three directions to move in, you must reconsider how to use the mouse buttons.

Due: Thursday October 26